Neutron with existing external network

Many people have asked how to use packstack –allinone with an existing external network. This method should allow any machine on the network to be able to access launched instances via their floating IPs. Also, at the end of this message, there are some ideas for making this process better that I thought we could discuss.

These instructions have been tested on Centos 7.

Initially, follow the Quickstart but stop when you see the first "packstack –allinone" at Step 3, instead do:

```
# packstack --allinone --provision-demo=n --os-neutron-ovs-bridge-mappings=extnet:br-ex --os-neutron-ovs-bridge-interfaces=br-ex:eth0 --os-neutron-ml2-type-drivers=vxlan,flat
```

This will define a logical name for our external physical L2 segment as "extnet". Later we will reference to our provider network by the name when creating external networks.

The command also adds 'flat' network type to the list of types supported by the installation. This is needed when your provider network is a simple flat network (the most common setup for PoCs). If you use a VLAN segment for external connectivity, you should add 'vlan' to the list of type drivers.

Note: the command is currently broken for Mitaka: https://bugzilla.redhat.com/show_bug.cgi?id=1316856, please skip –os-neutron-ovs-bridge-interfaces=br-ex:eth0 argument for now.

(There's an alternate method using packstack –allinone –provision-all-in-one-ovs-bridge=n, but it's more complicated)

After completion, given a single machine with a current IP of 192.168.122.212/24 via DHCP with gateway of 192.168.122.1:

Make /etc/sysconfig/network-scripts/ifcfg-br-ex resemble:

```
DEVICE=br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=static
IPADDR=192.168.122.212 # Old eth0 IP since we want the network restart to not
                       # kill the connection, otherwise pick something outside your dhcp range
NETMASK=255.255.255.0  # your netmask
GATEWAY=192.168.122.1  # your gateway
DNS1=192.168.122.1     # your nameserver
ONBOOT=yes
```

The file above will move the network parameters from eth0 to br-ex.

Make /etc/sysconfig/network-scripts/ifcfg-eth0 resemble (no BOOTPROTO!):

Note: if on Centos7, the file could be /etc/sysconfig/network-scripts/ifcfg-enp2s0 and DEVICE should be enp2s0

```
DEVICE=eth0
TYPE=OVSPort
DEVICETYPE=ovs
OVS_BRIDGE=br-ex
ONBOOT=yes
```

It is also possible to use a bond. In that case /etc/sysconfig/network-scripts/ifcfg-bond0 may look like this:

```
DEVICE=bond0
DEVICETYPE=ovs
TYPE=OVSPort
OVS_BRIDGE=br-ex
ONBOOT=yes
BONDING_MASTER=yes
BONDING_OPTS="mode=802.3ad"
```

This means, we will bring up the interface and plug it into br-ex OVS bridge as a port, providing the uplink connectivity.

Restart the network service

```
# reboot
```

or, alternatively:

```
# service network restart
```

Now, create the external network with Neutron.

```
# . keystonerc_admin
# neutron net-create external_network --provider:network_type flat --provider:physical_network extnet  --router:external
```

Please note: "extnet" is the L2 segment we defined with –os-neutron-ovs-bridge-mappings above.

You need to create a public subnet with an allocation range outside of your external DHCP range and set the gateway to the default gateway of the external network.

Please note: 192.168.122.1/24 is the router and CIDR we defined in /etc/sysconfig/network-scripts/ifcfg-br-ex for external connectivity.

```
# neutron subnet-create --name public_subnet --enable_dhcp=False --allocation-pool=start=192.168.122.10,end=192.168.122.20 \
                        --gateway=192.168.122.1 external_network 192.168.122.0/24
```

Get a cirros image, not provisioned without demo provisioning:

```
curl http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img | glance \
         image-create --name='cirros image' --visibility=public --container-format=bare --disk-format=qcow2
```

That's all you need to do from admin perspective to allow your users to connect their private networks to the outside world. Now let's switch to the user.

Since you haven't created a user yet:

```
openstack project create --enable internal
openstack user create --project internal --password foo --email bar@corp.com --enable internal
```

Now, let's switch to the newly created user:

```
# export OS_USERNAME=internal
# export OS_TENANT_NAME=internal
# export OS_PASSWORD=foo
```

Then create a router and set its gateway using the external network created by the admin in one of previous steps:

```
# neutron router-create router1
# neutron router-gateway-set router1 external_network
```

Now create a private network and a subnet in it, since demo provisioning has been disabled:

```
# neutron net-create private_network
# neutron subnet-create --name private_subnet private_network 192.168.100.0/24
```

Finally, connect your new private network to the public network through the router, which will provide floating IP addresses.

```
# neutron router-interface-add router1 private_subnet
```

Easiest way to the network and to launch instances is via horizon, which was set up by packstack.

See also

Watch this video for a demonstration of how to use DHCP on the bridge, including cloning the MAC address from eth0: https://www.youtube.com/watch?v=8zFQG5mKwPk